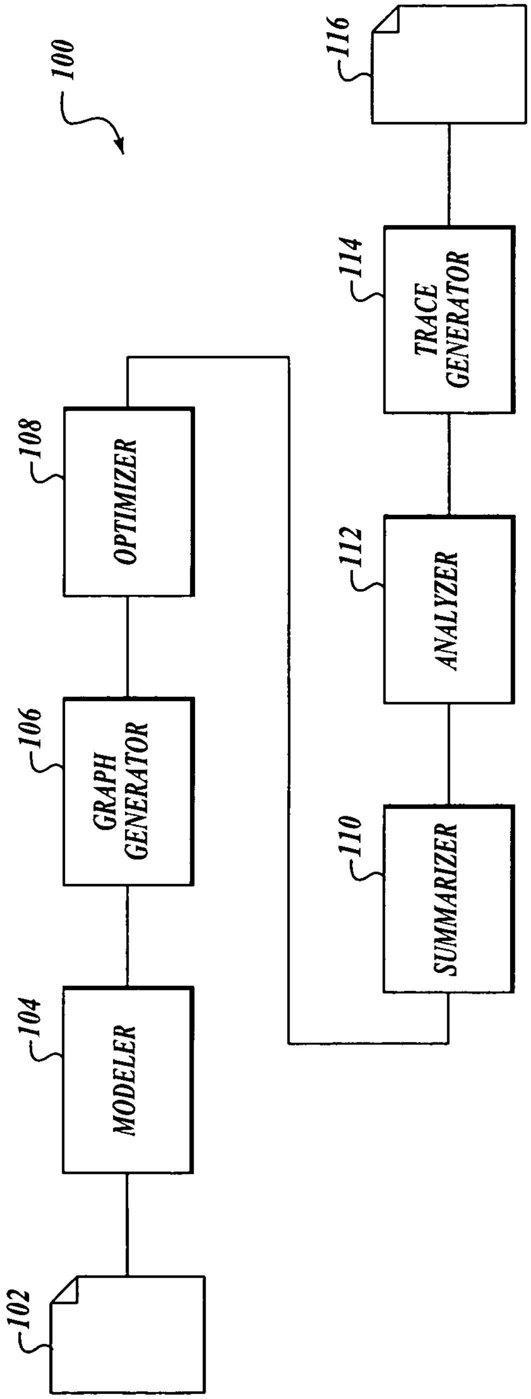
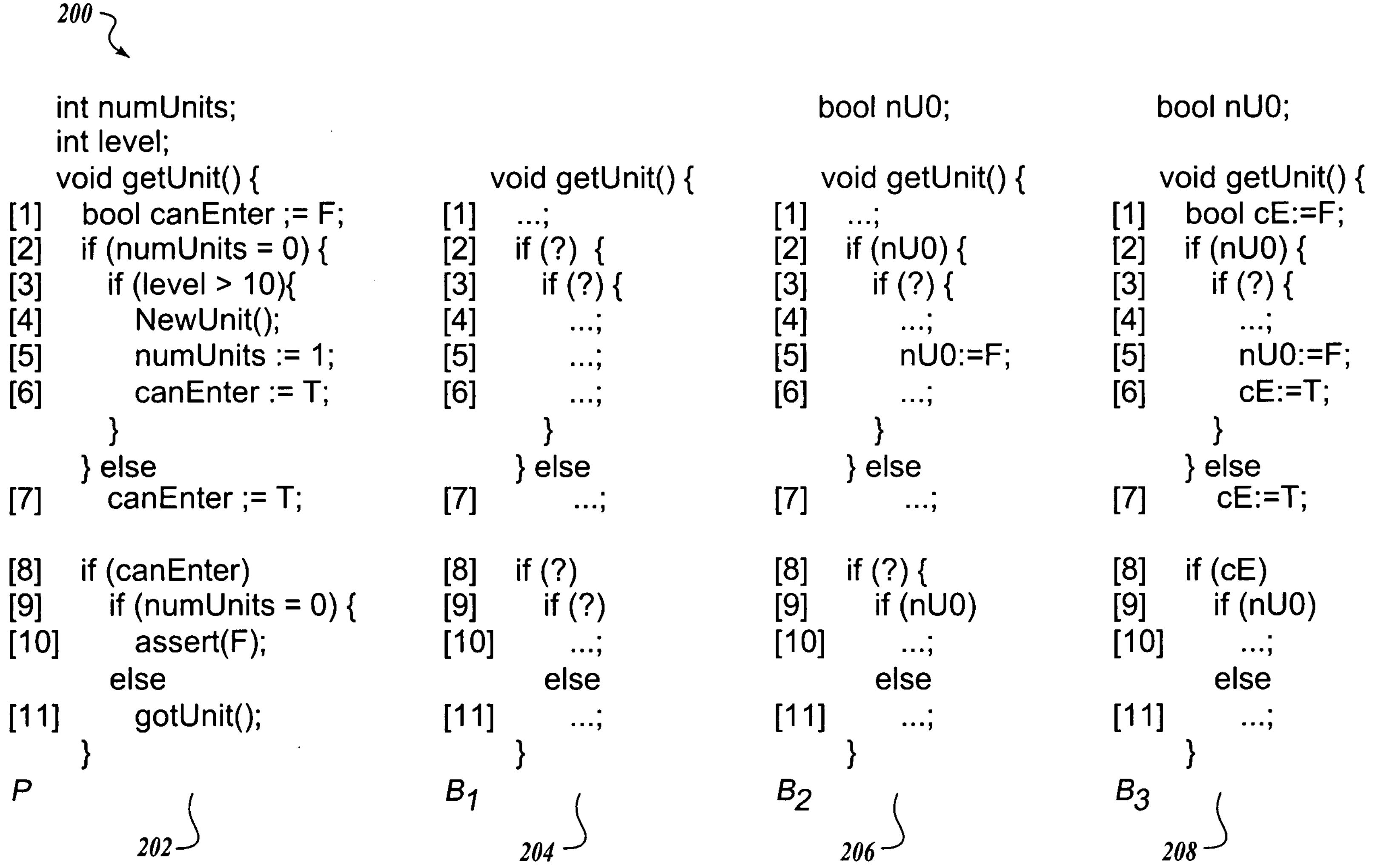

100
102
104
MODELER
106
GRAPH GENERATOR
108
OPTIMIZER
110
SUMMARIZER
112
ANALYZER
114
TRACE GENERATOR
116

FIG. 1



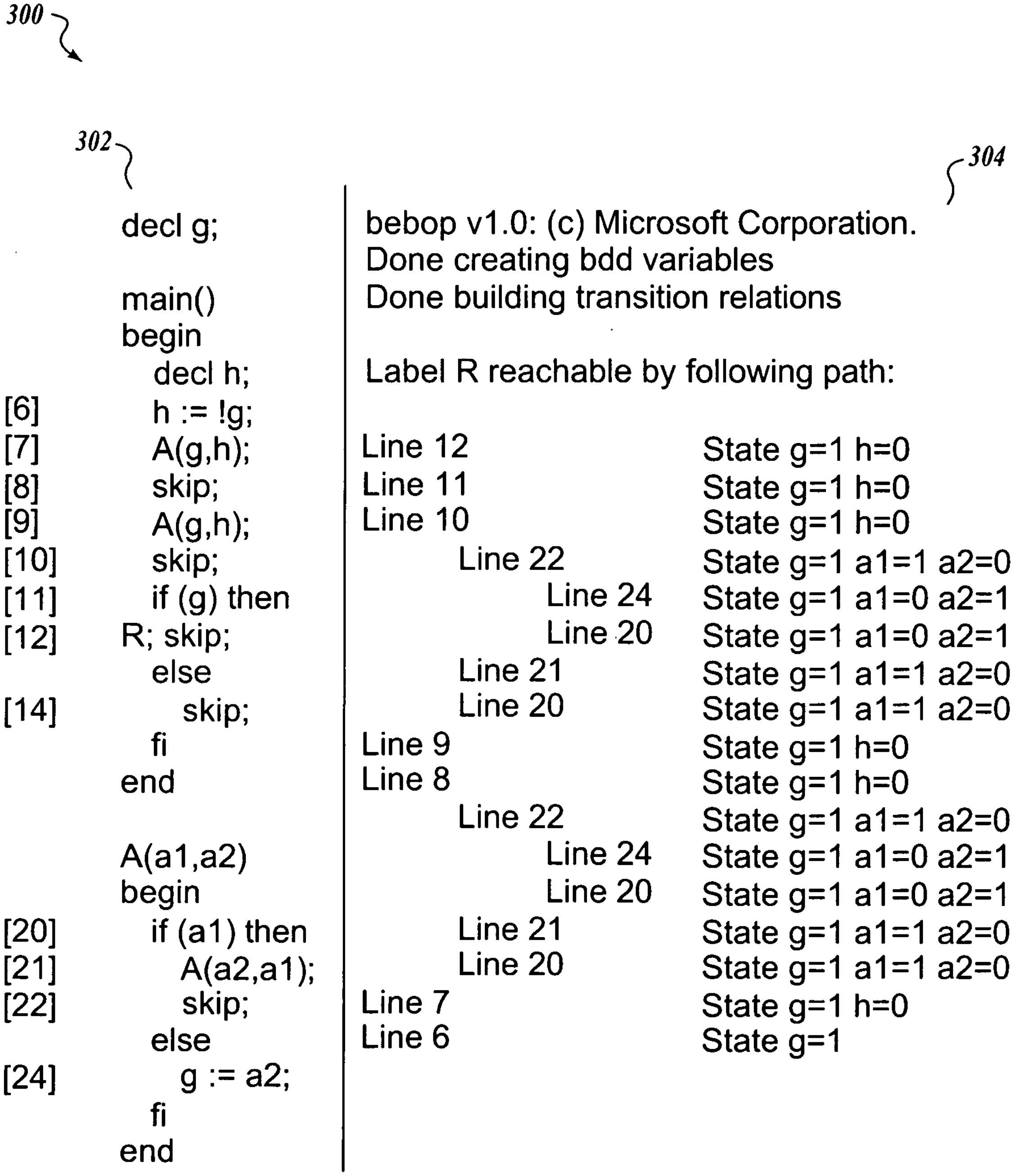
```
    int numUnits;
    int level;
    void getUnit() {
      bool canEnter ;= F;
      if (numUnits = 0) {
        if (level > 10){
          NewUnit();
          numUnits := 1;
          canEnter := T;
        }
      } else
        canEnter ;= T;

      if (canEnter)
        if (numUnits = 0) {
          assert(F);
        else
          gotUnit();
      }
```

```
    void getUnit() {
      ...;
      if (?)  {
        if (?) {
          ...;
          ...;
          ...;
        }
      } else
          ...;

      if (?)
        if (?)
          ...;
        else
          ...;
      }
```

```
    bool nU0;

    void getUnit() {
      ...;
      if (nU0) {
        if (?) {
          ...;
          nU0:=F;
          ...;
        }
      } else
          ...;

      if (?) {
        if (nU0)
          ...;
        else
          ...;
      }
```

```
    bool nU0;

    void getUnit() {
      bool cE:=F;
      if (nU0) {
        if (?) {
          ...;
          nU0:=F;
          cE:=T;
        }
      } else
        cE:=T;

      if (cE)
        if (nU0)
          ...;
        else
          ...;
      }
```

FIG. 2



```
        decl g;

        main()
        begin
          decl h;
[6]       h := !g;
[7]       A(g,h);
[8]       skip;
[9]       A(g,h);
[10]      skip;
[11]      if (g) then
[12]    R; skip;
          else
[14]        skip;
          fi
        end

        A(a1,a2)
        begin
[20]      if (a1) then
[21]        A(a2,a1);
[22]        skip;
          else
[24]        g := a2;
          fi
        end
```

```
bebop v1.0: (c) Microsoft Corporation.
Done creating bdd variables
Done building transition relations

Label R reachable by following path:

Line 12                 State g=1 h=0
Line 11                 State g=1 h=0
Line 10                 State g=1 h=0
     Line 22            State g=1 a1=1 a2=0
          Line 24       State g=1 a1=0 a2=1
          Line 20       State g=1 a1=0 a2=1
     Line 21            State g=1 a1=1 a2=0
     Line 20            State g=1 a1=1 a2=0
Line 9                  State g=1 h=0
Line 8                  State g=1 h=0
     Line 22            State g=1 a1=1 a2=0
          Line 24       State g=1 a1=0 a2=1
          Line 20       State g=1 a1=0 a2=1
     Line 21            State g=1 a1=1 a2=0
     Line 20            State g=1 a1=1 a2=0
Line 7                  State g=1 h=0
Line 6                  State g=1
```

FIG. 3

Inventor: Thomas J. Ball et al.
Docket No.: 50037.58US01
Title: METHODS FOR ENHANCING PROGRAM ANALYSIS
Serial No.: 09/866,090

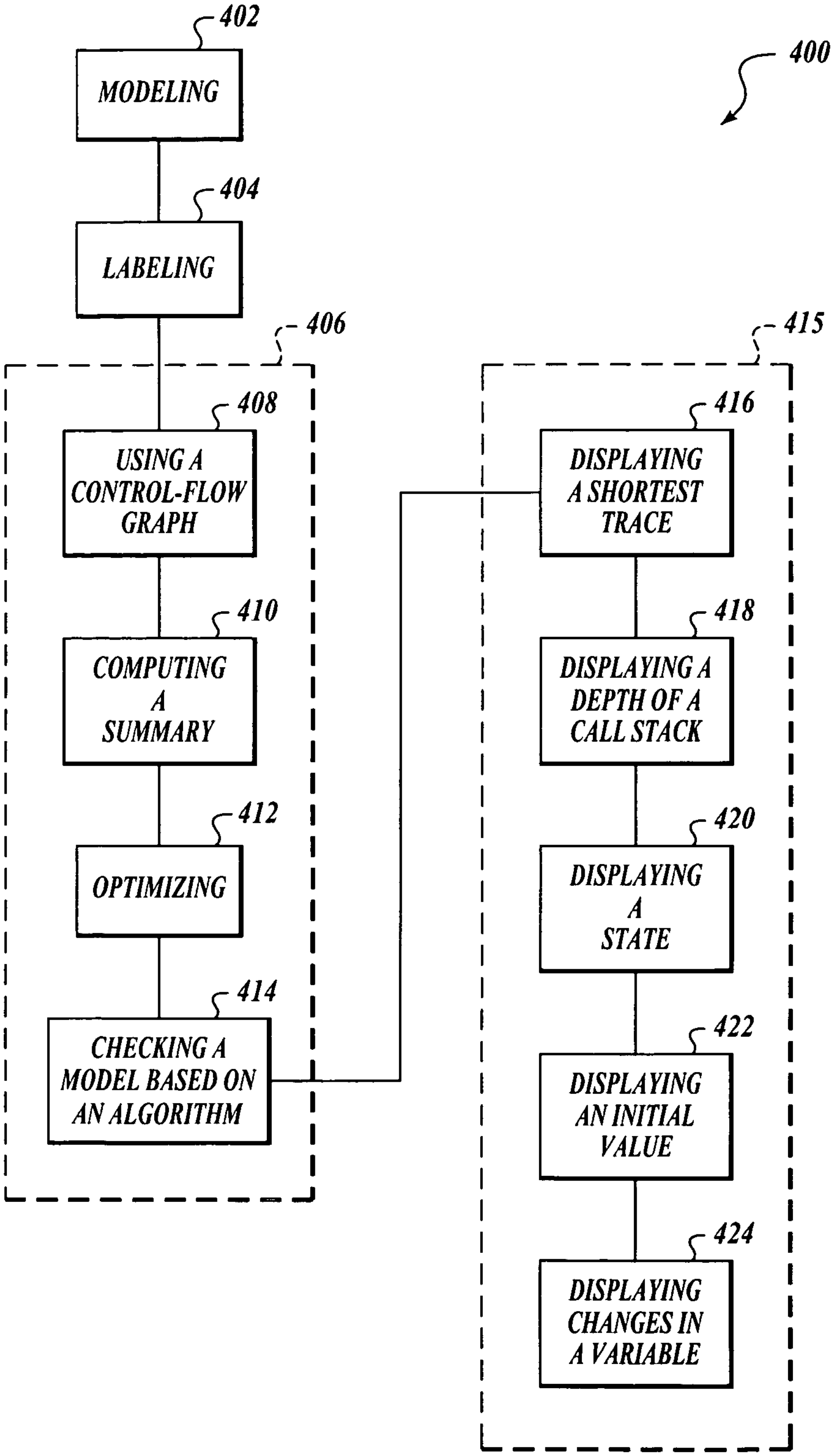

400
402
MODELING
404
LABELING
406
408
USING A CONTROL-FLOW GRAPH
410
COMPUTING A SUMMARY
412
OPTIMIZING
414
CHECKING A MODEL BASED ON AN ALGORITHM
415
416
DISPLAYING A SHORTEST TRACE
418
DISPLAYING A DEPTH OF A CALL STACK
420
DISPLAYING A STATE
422
DISPLAYING AN INITIAL VALUE
424
DISPLAYING CHANGES IN A VARIABLE

FIG. 4

500

```
[1]      if (z) {
[2]        x:=1;
         else
[3]        x:=z;
[4]      z:=y|x;
```

FIG. 5A

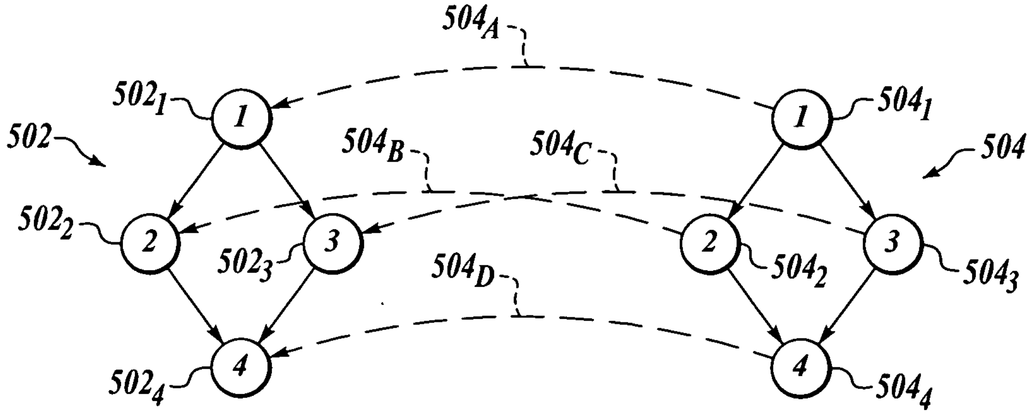


FIG. 5B

600

```
            Boolean g;
[1]         main() {
[2]            if (z) {
[3]               x:=1;
               else
[4]               x:=0;

[5]            z:=y+x;

[6]            foo (z);
[7]            skip;
[8]         }

[9]         foo (z) {
[10]           g:=1;
[11]        }
```

FIG. 6A

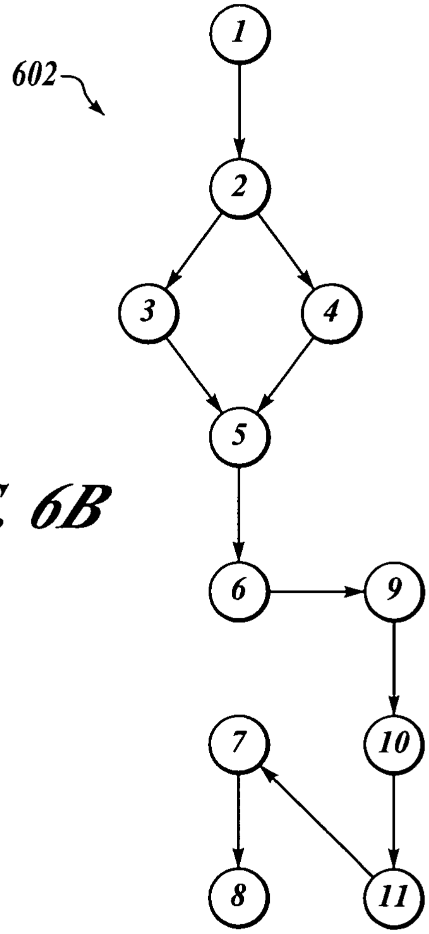


FIG. 6B

700

	v	$Transfer_v$
702	**skip** **print** **goto** **return**	$\lambda\langle\Omega_1,\Omega_2\rangle\cdot(\Omega_2=\Omega_1)$
704	$x_1,\ldots,x_k := e_1,\ldots,e_k$	$\lambda\langle\Omega_1,\Omega_2\rangle\cdot(\Omega_2=\Omega_1[x_1/\Omega_1(e_1)]\ldots[x_k/\Omega_1(e_k)])$

706	**if(*d*)**
	while(*d*)
	assert(*d*)
		$Transfer_{v,true}=\lambda\langle\Omega_1,\Omega_2\rangle\cdot((\Omega_1(d)=1)\wedge(\Omega_2=\Omega_1))$
		$Transfer_{v,false}=\lambda\langle\Omega_1,\Omega_2\rangle\cdot((\Omega_1(d)=0)\wedge(\Omega_2=\Omega_1))$

708	**pr**$(e_1,\ldots,e_k)$	$\lambda\langle\Omega_1,\Omega_2\rangle\cdot(\Omega_2=\Omega_1[x_1/\Omega_1(e_1)]\ldots[x_k/\Omega_1(e_k)])$, where $x_1,\ldots,x_k$are the formal parameters of pr

FIG. 7

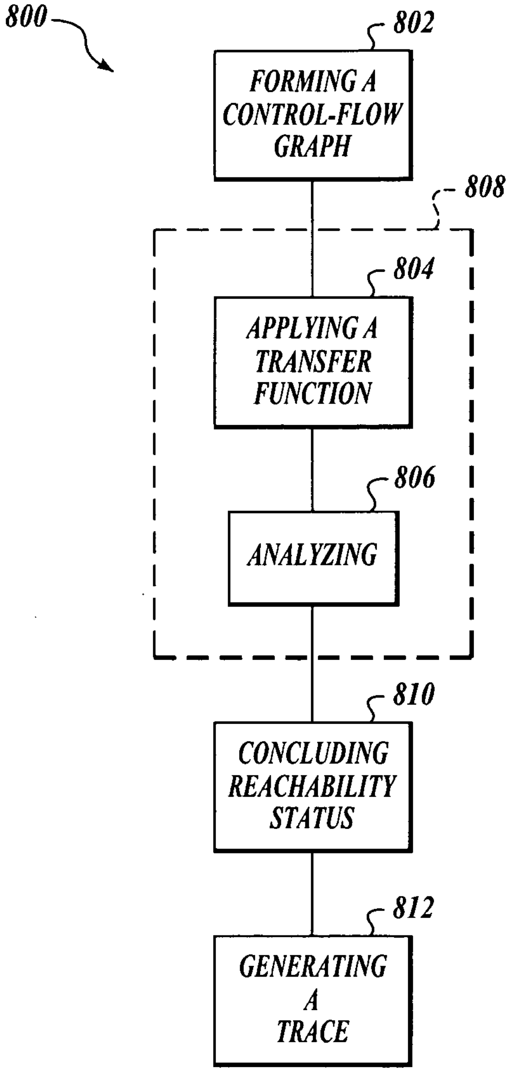

800
802
FORMING A CONTROL-FLOW GRAPH
808
804
APPLYING A TRANSFER FUNCTION
806
ANALYZING
810
CONCLUDING REACHABILITY STATUS
812
GENERATING A TRACE

FIG. 8

900

```
    global
902 $PathEdges, SummaryEdges, WorkList$

904 procedure Propagate($v,p$)
    begin
906   if p $\not\subset PathEdges(v)$ then
908     $PathEdges(v) := PathEdges(v) \cup$ p
910     Insert $v$ into $WorkList$ fi
      fi
912 end

914 procedure Reachable($G_B$)
    begin
916   for all $v \in V_B$ do $PathEdges(v) := \{\}$
917   for all $v \in Call_B$ do $SummaryEdges(v) := \{\}$
918   $PathEdges(First_B($main$)) :=$
        $\{\langle\Omega, \Omega\rangle \mid \Omega$ is any valuation to globals and local variables of main$\}$

920   $WorkList := \{First_B($main$)\}$
922   while $WorkList \neq 0$ do
924     remove vertex $v$ from $WorkList$
926     switch ($v$)
928       case $v \in Call_B$
            Propagate($_SSucc_B(v), SelfLoop(Join(PathEdges(v), Transfer_v))$) 930
            Propagate($ReturnPt_B(v), Join(PathEdges(v), SummaryEdges(v))$) 932
934       case $v \in Exit_B$:
            for each $w \in Succ_B(v)$ do 936
              let
                $c \in Call_B$ such that $w = ReturnPt_B(c)$ and 938
                $s = Lift_C(PathEdges(v), ProcOf_B(v))$ 940
              in
                if $s \not\subset SummaryEdges(c)$ then 944
                  $SummaryEdges(c) := SummaryEdges(c) \cup s$ 946
                  Propagate($w, Join(PathEdges(c), SummaryEdges(c))$); 948
              ni
950       case $v \in Cond_B$:
            Propagate($Tsucc_B(v), Join(PathEdges(v), Transfer_{v,true})$) 952
            Propagate($Fsucc_B(v), Join(PathEdges(v), Transfer_{v,false})$) 954
956       case $v \in V_B - Call_B - Exit_B - Cond_B$:
            let $p = Join(PathEdges(v), Transfer_v)$ in 958
              for each $w \in Succ_B(v)$ do 960
                Propagate($w,p$) 962
            ni
    end
```

FIG. 9

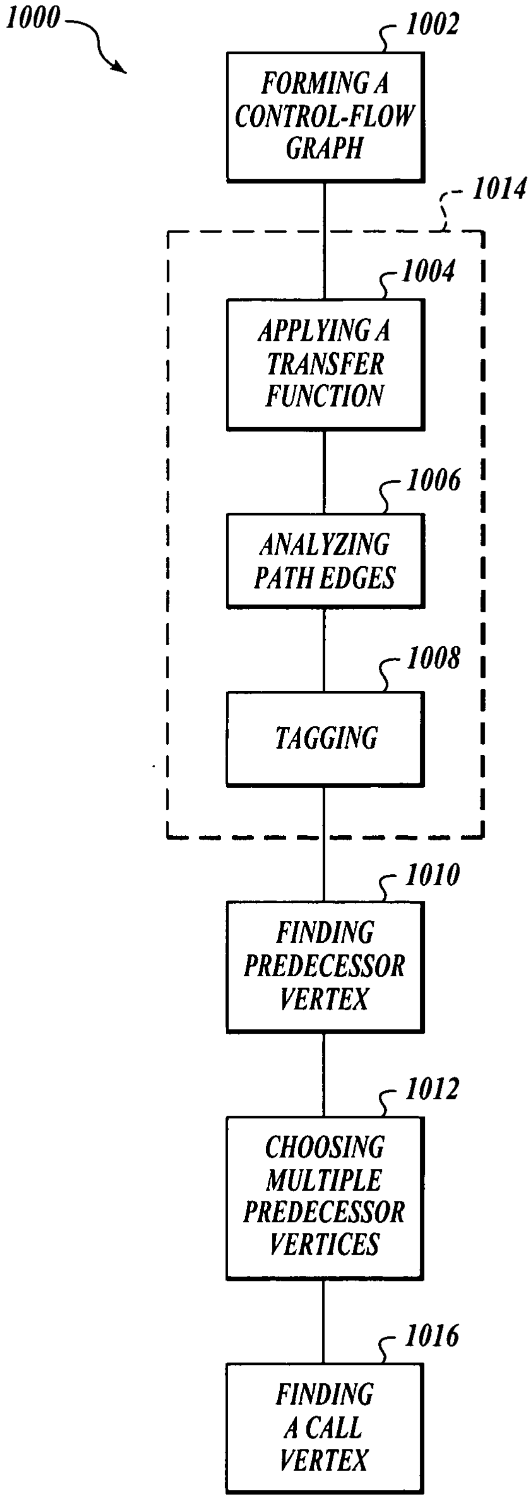

1000
1002
FORMING A CONTROL-FLOW GRAPH
1014
1004
APPLYING A TRANSFER FUNCTION
1006
ANALYZING PATH EDGES
1008
TAGGING
1010
FINDING PREDECESSOR VERTEX
1012
CHOOSING MULTIPLE PREDECESSOR VERTICES
1016
FINDING A CALL VERTEX

FIG. 10

Inventor: Thomas J. Ball et al.
Docket No.: 50037.58US01
Title: METHODS FOR ENHANCING PROGRAM ANALYSIS
Serial No.: 09/866,090

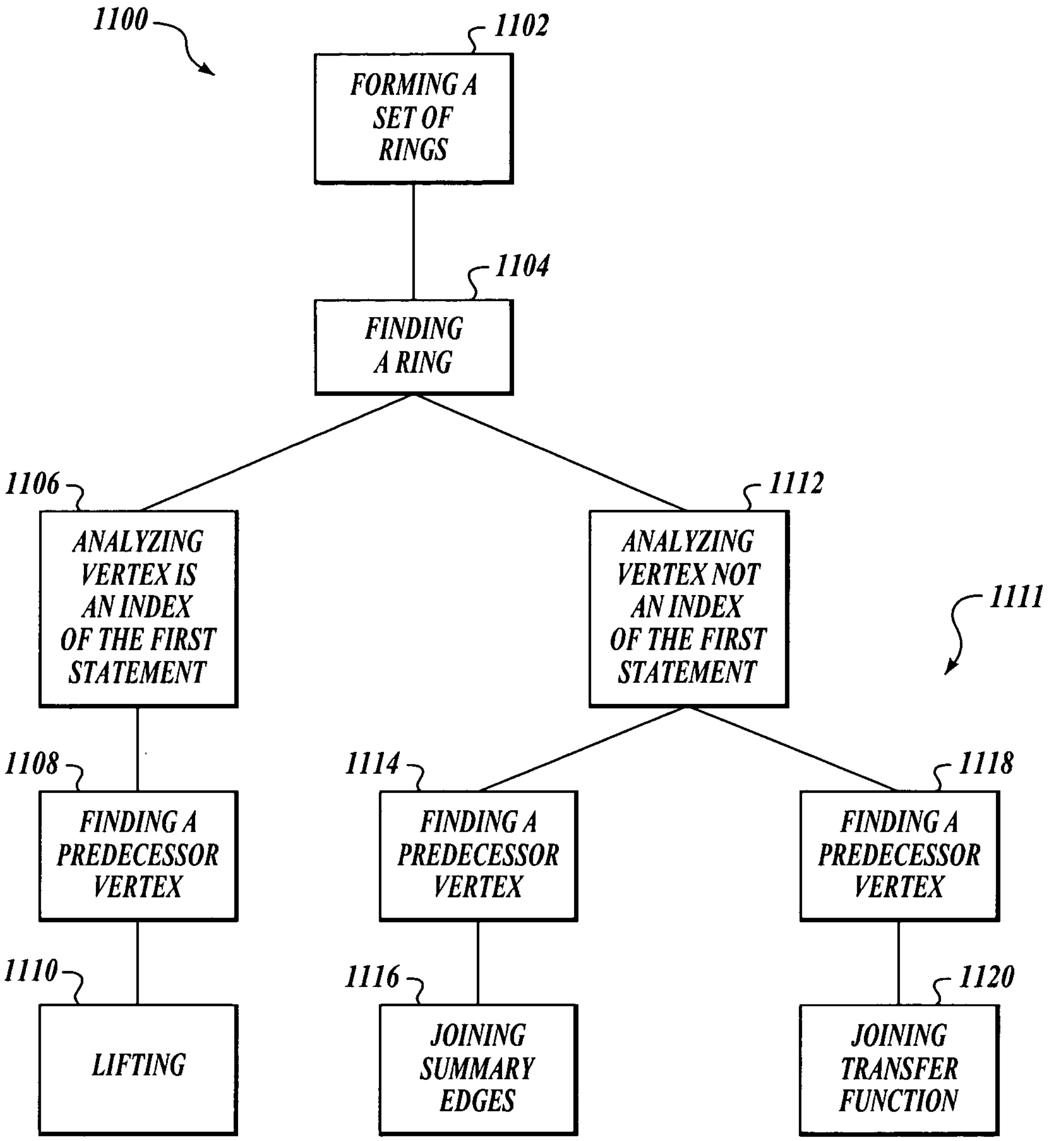

1100
1102
FORMING A SET OF RINGS
1104
FINDING A RING
1106
ANALYZING VERTEX IS AN INDEX OF THE FIRST STATEMENT
1112
ANALYZING VERTEX NOT AN INDEX OF THE FIRST STATEMENT
1111
1108
FINDING A PREDECESSOR VERTEX
1114
FINDING A PREDECESSOR VERTEX
1118
FINDING A PREDECESSOR VERTEX
1110
LIFTING
1116
JOINING SUMMARY EDGES
1120
JOINING TRANSFER FUNCTION

FIG. 11

```
global
$PE' : V_G \rightarrow$ set-of (D x D)
$Worklist\ V_G \rightarrow$ set-of (D x D)

procedure Propagate($v : V_{G,p} :$ (D x D))
begin
  if $p \notin PE'(v)$ then
      $PE'(v) := PE'(v) \cup \{p\}$
      $Worklist(v) := Worklist(v) \cup \{p\}$
  fi
end

procedure $CMOP_{SP_{rhs}}$(S : set-of D)
begin
  $PE'(entry) := \{\langle d, d \rangle \mid d \in S\}$
  $Worklist(entry) := PE'(entry)$
  while $\exists v_2$ s.t $Worklist(v_2) \neq 0$ do
    select and remove $\langle d_1, d_2 \rangle$ from $Worklist(v_2)$
    for each $v_2 \rightarrow v_3 \in E_G$ do
      for each $d_3 \in M(v_2 \rightarrow v_3)(\{d_2\})$ do
        Propagate($v_3 \langle d_1, d_3 \rangle$)
      od
    od
  od
end
```

FIG. 12

```
global
$PE' : V \rightarrow$ set-of (set-of D x set-of D)
$Worklist : V_G \rightarrow$ set-of (set-of D x set-of D)

procedure Propagate($v : V_{G,p}$ : (set-of D x set-of D))
begin
  if $p \notin PE'(v)$ then
     $PE'(v) := PE'(v) \cup \{p\}$
     $Worklist(v) := Worklist(v) \cup \{p\}$
  fi
end

procedure $CSMOP_{SP_{rhs}}$(S' : set-of (set-of D))
begin
   $PE'(entry) := \{\langle S, S\rangle \mid S \in S'\}$
     $Worklist(entry) := PE'(entry)$
     while $\exists v_2$ s.t $Worklist(v_2) \neq 0$ do
       select and remove $\langle S_1, S_2\rangle$ from $Worklist(v_2)$
       for each $v_2 \rightarrow v_3 \in E_G$ do
         let $S_3 = M(v_2 \rightarrow v_3)(S_2)$ in
           Propagate($v_3\langle S_1, S_3\rangle$)
         ni
       od
     od
   end
```

FIG. 13